

## 轻松话题（二） | 给你分享我的工作百宝箱

2020-05-25 罗剑锋

罗剑锋的C++实战笔记

[进入课程 >](#)



讲述: Chrono

时长 11:05 大小 10.16M



你好，我是 Chrono。

今天，我再来说点 C++ 之外的话题。我来聊聊我的工作方式，给你介绍一下我用的开发环境，有快捷键、配置脚本、常用命令什么的。算不上什么“高效技巧”，但是也能从小处提高工作效率，希望能给你一点借鉴。

### Linux

我主要是在 Linux 上写程序，经常要登录到内部的服务器，但我常用的笔记本或者台式机上装的还是 Windows，所以需要有一个“趁手”的客户端。



Windows 上有很多这样的软件，最早我用的是 PuTTY，但其他很多同事用的是 XShell。不过，现在的我已经都不用这些了。

你一定想知道，难道还有什么比 PuTTY、XShell 更好更强大的远程终端吗？

要说有也算有，要说没有也算是没有。

因为，现在我就把 Linux 操作系统当成终端机来使用，就用它内置的 Terminal、ssh 命令来实现远程登录。

具体的做法也很简单，安装一个 VirtualBox，再最小化安装一个 Ubuntu，就可以了。

这么做的好处在哪里呢？

首先，这个环境完全是免费的，不需要注册或者破解。其次，它本身就是 Linux，与开发环境相同，可以用来在本地做试验、“练手”。再次，Linux 里有非常丰富的工具可以下载安装，能够随心所欲地定制环境，用起来非常舒心。

当然，把 Linux 转换成一个高效的终端，还是需要一点点“技巧”的，接下来，我就跟你说说我的做法，要点就是“**全程键盘操作**”。

第一个，用“Ctrl+Alt+T”可以直接打开命令行窗口，而不必用鼠标去点图标，然后用“Ctrl+Shift+T”可以开新标签页，这样就可以很方便地实现多窗口登录，不会像某些软件那样有数量的限制。

另外，我选择的是 Ubuntu 14.04，在这个版本里，可以用鼠标右键点标签页直接改标题名，区分不同的窗口，即使开多个标签也可以轻松管理（但这个功能在后来的 16.04、18.04 却给去掉了，只能额外写 Shell 脚本来实现，有那么一点不爽）。

第二个，修改 Shell 的配置文件“.bashrc”或者是“.profile”，在里面加上一行“set -o vi”。

```
1 #.bashrc
2 set -o vi
```

 复制代码

这样，你就可以在命令行里实现 vi 操作了，按一下 ESC 键，就进入到了 vi 模式，可以用 “/” 快速查找之前的历史命令，而不必每次都要敲完整的命令。

比如说，之前输入了一条命令 “ssh chrono@10.1.1.25” 登录服务器，那么，下次再登录时就没有必要再敲一遍了，只要按 ESC，然后输入 “/25”，回车，Linux 就可以帮你找到上次的这条命令。这时，你就可以轻松愉快地登录了。

用 Linux 作为终端的唯一一个缺点，是它无法自动填写登录密码，每次都要手动敲，这个的确比较烦人。所以，只能把登录密码尽量改得简单好输入，最好是键盘上的固定模式（比如设置成 “qazwsx”），这样就可以在 1 秒内完成。

## Vim

写代码就要用到编辑器，在 Windows 里，常用的有 VS Code、Sublime，等等，而在 Linux 里，最佳的选择可能就是 Vim 了。

说是 Vim，但我更愿意称之为 vi。一方面是早期的使用习惯（我最早用的是 AIX，上面只有 vi，而不是 Vim），另一个更重要的原因是可以少打一个字符。可不要小看了这一点效率的提升，想想每天你要说多少次、用多少次 vi 吧。

有的人可能还是习惯在 Windows 上的编辑器里写代码，然后通过某种方式上传到 Linux，再编译和运行。我个人觉得这种做法不太可取，既然是 Linux 开发，就应该全程在 Linux 上工作，而且很多时候会现场调试，不可能有那么合适的编辑器。

所以，尽早抛弃 “窗口 + 鼠标” 式编辑器，强迫自己只用 vi，就可以尽快熟悉 vi 的各种操作，让你在 Linux 上 “运指如飞”。

另外，你可能知道，vi 也有很多的插件，比如 ctags，搭配上众多的插件会让 vi 更 “现代化”。但对于服务器开发来说，还是那个问题：不是每台服务器都会给你配置得那么完善的。与其倒腾那些 “花里胡哨” 的插件，不如 “离开舒适区”，练好 vi 的基本功，到哪里都能吃得开。

最基本的 vi 操作，我就不多谈了，网上一搜一大堆，我来说几个写代码时比较实用的命令。

1. “:tabnew” ，新建一个编辑窗口，也就是支持多标签操作，多个标签可以用 “gt” 切换。

2. “Ctrl+V” “Shift+V” 的整列整行选择，然后就可以用 “x” 剪切、“p” 粘贴。

“Ctrl+V” 的列选择功能还有一个衍生的方便技巧：选择多列后按 “I” ，再输入 “/” ，按 ESC，就可以在每行前面都插入 “/” ，轻松地实现大段代码的工整注释。

3. “Ctrl+P” 是 vi 内置的 “代码补全” 功能，对我们程序员来说特别有用。只要写上开头的一两个字符，再按 “Ctrl+P” ，vi 就可以提示出文件里曾经出现的词，这样，在写长名字时，就再也不用害怕了。

不过，vi 的 “代码补全” 功能还是比较弱的，不是基于语法分析，而是简单的文本分词，但我们也不能太苛求。

4. 可以随时用 “Ctrl+Z” 暂停 vi，把它放到后台，然后执行各种 Shell 操作，在需要的时候，只要敲一个 “fg” 命令，就可以把 vi 恢复回来。

这在调试的时候非常有用，改改代码，运行一下，看看情况再切回来继续改，不用每次重复 vi 打开源文件，而且可以保留编辑的 “现场” 。

除了刚才的这四点操作技巧，想要用好 vi，还必须要对它做适当的配置，比如显示行号、控制缩进，等等。下面就是我常用的 “.vimrc” ，非常短小，基本上我每登录一台新服务器，就会把这个配置复制过去，这样，无论在哪里，vi 都会是我熟悉的环境。

 复制代码

```
1 #.vimrc
2 set nu
3 sy on
4 set ruler
5 set smartindent shiftwidth=4
6 set tabstop=4
7 set expandtab
8
```

```
9 set listchars=tab:~>,trail:~
10 set list
11
```

## Git

写完了程序，还要用适当的版本控制系统把它管理起来，否则源码丢失、版本回溯、多人协作等问题会把你弄得焦头烂额。

我最早用的是微软的 VSS (Visual Source Safe) ，后来用过 IBM 的 ClearCase，再后来又用 SVN，现在则是 Git 的“铁杆粉丝”。

Git 的好处实在太多了：分布式、轻量级、可离线、开分支成本低.....还有围绕着它的 GitHub/GitLab 等高级团队工作平台，绝对是最先进的版本控制系统。


如果在 2020 年的今天，你所在的公司还在用 SVN 这样的“上古”软件，可真的是要考虑一下项目的前景了。

Git 有许多高级用法，有的也很复杂，我不可能也没必要把那些讲清楚。所以，我只介绍一个能够简化 Git 操作的小技巧：**命令别名**。

Git 的命令含义明确，但缺点是单词太长，多次操作输入就显得很繁琐，这点就不如 SVN 命令那么简单明了。好在我们可以在 Git 的配置文件“.gitconfig”里为这些命令起别名，比如把“status”改成“st”，把“commit”改成“ci”。

下面这个就是我常用的一个 Git 配置，里面还有个特别的地方是在“diff”的时候使用“vimdiff”，用可视化的方式来比较文件的差异，比原始的“diff”更好。

```
1 [alias]
2 st = status
3 ci = commit
4 br = branch
5 co = checkout
6 au = add -u .
7 ll = log --oneline --graph
8 d = difftool
9 [diff]
10 tool = vimdiff
```

 复制代码

## GDB

最后来说一下调试工具 GDB 吧，它应该是 Linux 程序员最得力的一个帮手了。

标准的 GDB 是纯命令行式的，但也有一些基于它的图形化工具（比如 DDD、Data Display Debugger），但用好 GDB 命令行调试，还是我们的一项基本素质。

### **GDB 不仅是一个调试工具，它也是一个学习源码的好工具。**

单纯的源码是静态的，虽然你可以分析它的整体架构，在头脑里模拟出它的工作流程，但计算机实在是太复杂了，内外部环境因素很多，仅靠“人肉分析”很难完全理解它的逻辑。

这个时候，GDB 就派上用场了，以调试模式启动，任意设定外部条件，从指定的入口运行，把程序放慢几万倍，细致地观察每个变量的值，跟踪代码的分支和数据的流向，这样走上几个来回之后，再结合源码，就能够对程序的整体情况“了然于胸”。

GDB 用得久了，差不多每个人都会有一些自己的心得。我列出一些我觉得能够提高调试效率、最有价值的命令。

pt: 查看变量的真实类型，不受 typedef 的影响。

up/down: 在函数调用栈里上下移动。

fin: 直接运行到函数结束。

i b: 查看所有的断点信息。

wh: 启动“可视化调试”。这个是最喜欢的命令，可以把屏幕分成上下两个窗口，上面显示源码，下面是 GDB 命令输出，不必再用“l”频繁地列出源码了，能够大大提高调试的效率。

## 小结

好了，今天的话题就到这里，简单小结一下我的工作环境，给你一个参考：

1. 我选择 Linux 虚拟机作为登录服务器的终端，可以很容易开多窗口操作；

2. 我选择 Vim 作为编辑器，熟记常用命令后写代码也很方便；
3. 我选择 Git 作为版本管理工具，使用别名来简化命令；
4. GDB 是调试 C++ 程序的利器，也可以用来学习源码。

那么，你是否也有一些工作中的实用小技巧呢？欢迎一起来分享。



# 课外小贴士

1. 使用Linux虚拟机时，最好再设置一下本地的“共享目录”，方便上传下载各种资料（使用scp）。
2. 也有一些方法可以让ssh自动填充密码，例如使用expect，但我觉得有点麻烦，所以一直没用。
3. 在“.bashrc”里，还可以用“alias”命令设置常用Shell命令的别名，比如，我常用的有“alias l='ls -lh'”“alias vi=vim”，也可以提高工作效率。
4. 直接用root身份登录服务器是一个危险的操作，虽然很多人都这么做。我的建议是创建一个普通用户，然后用visudo赋予sudo权限，只在必须用root权限的时候才用sudo。
5. GCC/LLVM等著名编译器都已经相继从SVN迁移到了Git。



# 6月-7月课表抢先看

## 充 ¥500 得 ¥580

赠「¥ 118 月球主题 AR 笔记本」



【点击】图片, 立即查看>>>

© 版权归极客邦科技所有, 未经许可不得传播售卖。页面已增加防盗追踪, 如有侵权极客邦将依法追究其法律责任。

上一篇 轻松话题 (一) | 4本值得一读再读的经典好书

### 精选留言 (17)

写留言



嵇斌

2020-05-25

看了这篇发现就四个字: 朴实无华。

展开

作者回复: 老环境用习惯了, 见笑。



6



张JL

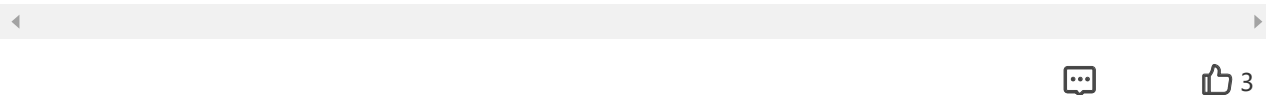
2020-05-26

罗老师的这个课程看到现在, 给我最大的感觉就是接地气, 说的很细致, 也很实用。干货满满, 但是又觉得努把力自己也能做的到, 不会有可望不可及之感。

展开 ▾

作者回复: 嗯, 因为我自己就是从小白过来的, 现在也一直是在第一线, 所以说的就都是自己的亲身体会, 实用至上。

如果所有学这个课程的同学都有这种感受就太好了。



**lckfa李钊**

2020-05-25

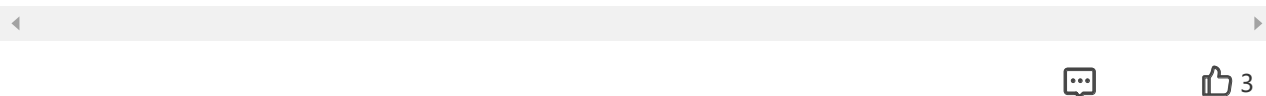
现在使用Windows10 的wsl2, 即Linux子系统, 可以拥有在Linux下一样的开发体验, 其原理应该也是Linux虚拟机。不得不说, 微软在这方面的进步有目共睹。工具的话, 我使用VSCode, sublime和Windows Terminal。

vi之前用过一段时间, 但是手残党, 现在只记得:!qw了。看老师用的这么溜, 又心痒痒了。Git确实是个极好的版本工具, 自从用了git, 我的coding life变得舒服多了。...

展开 ▾

作者回复: 微软的形象这些年确实在改变, 不再是“业内公敌”, 只是我现在是很难改用回Windows了。

GitHub是所有程序员都必须用的网站, 实在的好东西。



**Eric**

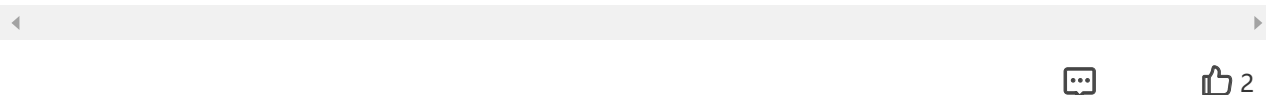
2020-05-28

一直以来在Linux环境开发, 从没用过IDE, 都是一套vim走天下, 很认同全键盘提高效率的理念。自己也做了一套vim配置放在GitHub上, 有不少同事在用。分享一下参考: <https://github.com/sky8336/skyVim>

每次在一台新的机器上, 执行一两个脚本就配置好了自己熟悉的东西。还是很方便的。

展开 ▾

作者回复: 欢迎同好交流经验心得。



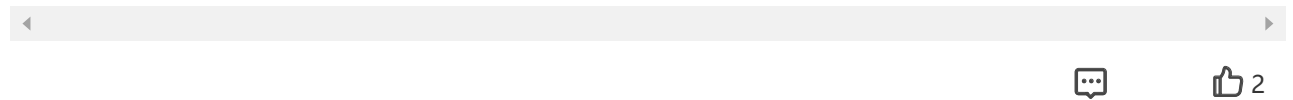
**风华神使**

2020-05-25

我直接windows terminal, 别的ssh客户端都卸载了

展开 ▾

作者回复: 好同学, 有魄力, 笑。



**qinsi**

2020-05-27

多窗口: tmux

命令行历史: hstr

免密登录: ssh-copy-id

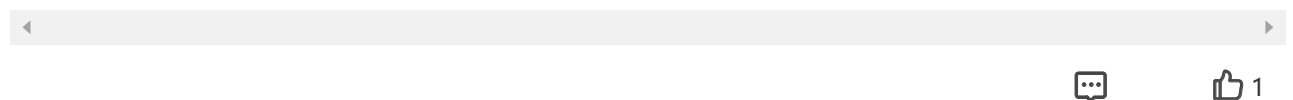
diff: delta

作者回复: 这些也都是很不错的工具, 很好的分享。

tmux我以前用过, 操作略麻烦了点, 后来就没怎么用。

hstr需要安装, 而vi也可以达到类似的效果。

后两个没用过, 有机会试一下, 非常感谢。



**文超**

2020-05-25

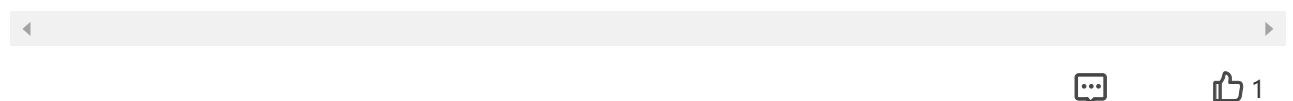
老师讲得这些都偏向于后台开发方向的, 对于客户端程序, 部分观点实在不能苟同。

对于底层图形学开发、引擎类的开发, 还是非常鼓励使用IDE集成开发环境的, 能够极大的提升开发体验。

作者回复: 因为我一直以来都是后端开发, 所以只能讲这方面的经验了。

图形学、引擎类的没做过, 没有发言权, 就不多说了。

欢迎有IDE环境经验的同学分享。



**silverhawk**

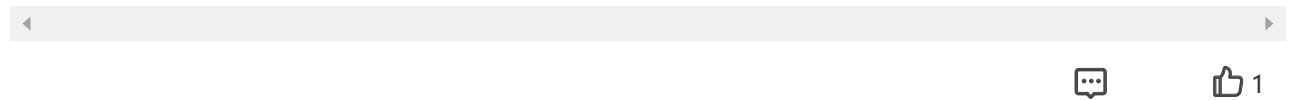
2020-05-25

Windows Subsystem for Linux Installation,最近build大会才进一步原生支持, 可以直接

VSCode在window下做IDE，但是其他操作环境在Linux下

展开 ▾

作者回复: win10用的少，一直虚拟机用习惯了，各种场合都适用。

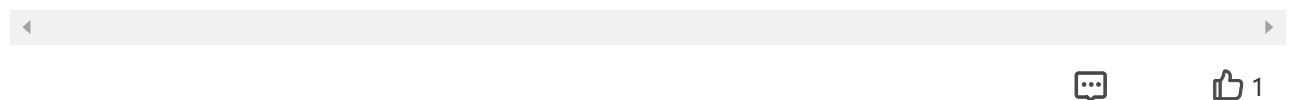


泡泡龙

2020-05-25

VBox的增强工具插件是个头疼的问题，我现在安了四个发行版了，只要虚拟机里的Linux系统一更新，比如内核软件包更新后，视频分辨率就不对了，就没法自适应窗口大小了。老师有没有遇见过这种问题？

作者回复: virtualbox确实有一些问题，我在mac上用的时候分辨率就挺烦人，更新版本也没有完全解决。可惜的是只有这么一个免费的可以用。

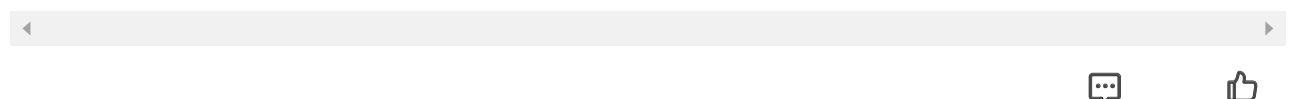


jxon-H

2020-05-27

非常称手的“兵器”，谢谢罗老师，我收下了，我也要拿它来练“武功”。

作者回复: 有了心得后欢迎再来分享。



锦鲤

2020-05-27

罗老师，请教下，比较初级的问题，跟文中的主题没有关联。Linux下的<linux/types.h>这个头文件，大致内容：

```
#ifndef _LINUX_TYPES_H
#define _LINUX_TYPES_H
#include <asm/types.h>...
```

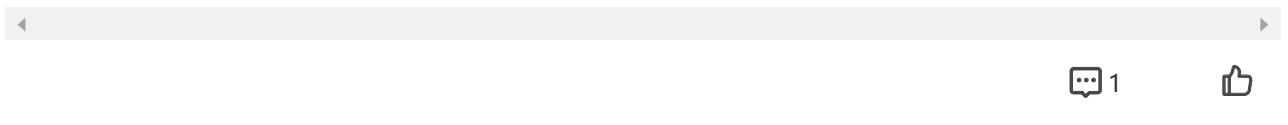
展开 ▾

作者回复: 很久没有在Windows上开发了，情况不是太清楚，只能大略说一下。

linux/types.h头文件应该是系统相关的，在Windows上大概率是没有完全对应的，但我们可以自己做跨平台兼容。

比如先用条件编译，检查系统，再包含头文件。对于必须的类型，可以用宏定义来屏蔽系统差异。

这方面可以参考一下Nginx源码，它是跨平台的，去GitHub上看看os目录里是如何做的吧。



**java2c++**

2020-05-27

ssh密码设置简单终究需要输入密码，其实有免密登陆的，命令是ssh-key-gen生成公钥私钥，把公钥copy到目标机就可以了的。

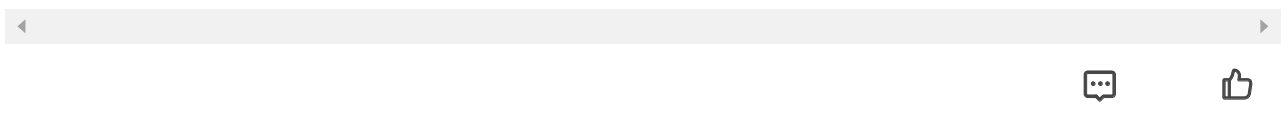
使用虚拟机的优势我没有看明白呢，毕竟在个人电脑上装虚拟机很耗资源，xshell远程登录到目标机后一样可以使用vim，设置alias啊

展开 ▾

作者回复:

1.这是个方法，可能是惯性思维了，以前一直没想到。

2.xshell好像是未注册版开多窗口有限制，而且Linux就可以开多个窗口，而且即使不登录也可以用很多工具，比如man。



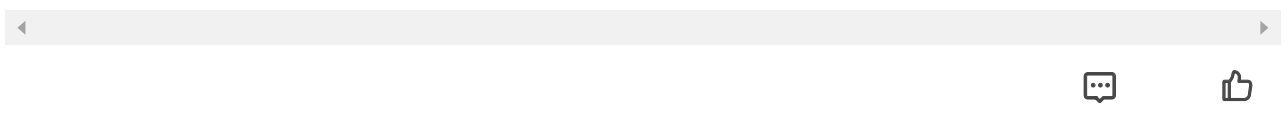
**SometimesNever**

2020-05-26

这个gdb命令太实用了，👍👍👍👍

展开 ▾

作者回复: gdb比较复杂，很多东西我也没有完全掌握，你如果有觉得好用的命令欢迎分享出来。



**完全不会C++**

2020-05-25

受益良多,继续跟着老师前进

展开 ▾

作者回复: 有自己的好经验也欢迎一起分享。



文若

2020-05-25

老师，我们公司的项目主要中针对传统企业定制项目，一直使用的是SVN 管理代码。使用的方法是建立一个基础分支，然后针对每个项目建立分支，升级版本再在分支上建立分支。目前维护起来很麻烦，经常修改一个问题，要在的多个分支上修改，切换到git是否能好一些？

展开 ∨

作者回复: 当然了，svn很重，开分支成本非常高，不适合现在的快速开发流程了。

git的分支非常轻，几乎没有成本，所以可以随便开分支试验或者改bug，然后任意合并修改。

有本电子书pro git，里面写的很详细，也可以找找其他资料，看一天应该就能掌握了。



靳远东

2020-05-25

都说Docker比虚拟机好用，老师用过没有

展开 ∨

作者回复: 抱歉啊，没怎么用过，帮不上你。



吃鱼

2020-05-25

我也是vim深度依赖 但是现在更喜欢vscode装上vim插件远程连接 既照顾了手指习惯又获得了好的补全和提示体验

展开 ∨

作者回复: vscode我也装了vi插件，用来看代码还是很不错的，但用惯了unix，windows 界面总是不太适应。





